GB2363286

Publication Title:

Network managing system

Abstract:

A network managing system comprising a configuration management information schema definition storage unit for storing therein a configuration management information schema definition describing classes representing apparatuses in a network and failure events observed in the classes, a propagation model storage unit for storing therein propagation models between failure events, a configuration information storage unit for storing therein configuration information about the apparatuses actually installed in the network, and an inferring device for making inference based on the configuration management information schema definition, the propagation models, and the configuration information,; including a counter for dealing with the failure events occurring in the apparatuses and a comparator for inferring the causes of the failure events based on the contents of the counter according to a predetermined method.

(12) UK Patent Application (19) GB (11) 2 363 286 (13) A

(43) Date of Printing by UK Office **12.12.2001**

(21) Application No **0114407.0**

(22) Date of Filing **28.07.1999**

(86) International Application Data
PCT/JP99/04041 Jp 28.07.1999

(87) International Publication Data
WO01/08016 Jp 01.02.2001

(71) Applicant(s)
Sumitomo Electric Industries, Ltd.
(Incorporated in Japan)
5-33, Kitahama 4-chome, Chuo-ku, Osaka, Japan

(72) Inventor(s)
Noriaki Kuwahara

(74) Agent and/or Address for Service
Boult Wade Tennant
Verulam Gardens, 70 Gray's Inn Road, LONDON, WC1X 8BT, United Kingdom

(51) INT CL^7
G06F 13/00 11/22 , H04L 12/24

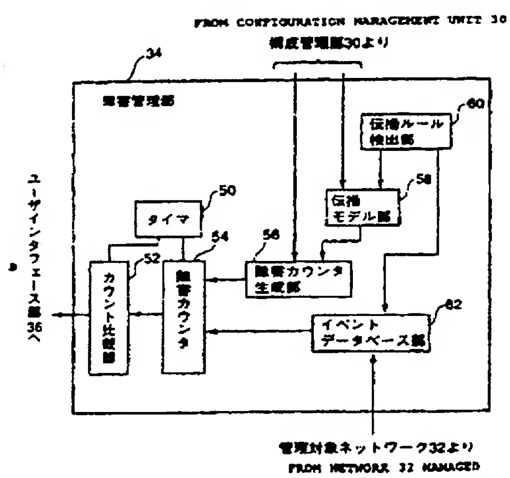
(52) UK CL (Edition S)
H4K KFMA

(56) Documents Cited by ISA
JP 090247145 A JP 080018593 A JP 070030540 A

(58) Field of Search by ISA
INT CL^6 **G06F 13/00**
JITSUYO SHINAN KOHO 1922-1996 ;
TOROKU JITSUYO SHINAN KOHO 1994-1999 ;
KOKAI JITSUYO SHINAN KOHO 1971-1999 ;
JITSUYO SHINAN TOROKU KOHO 1996-1999

(54) Abstract Title
Network managing system

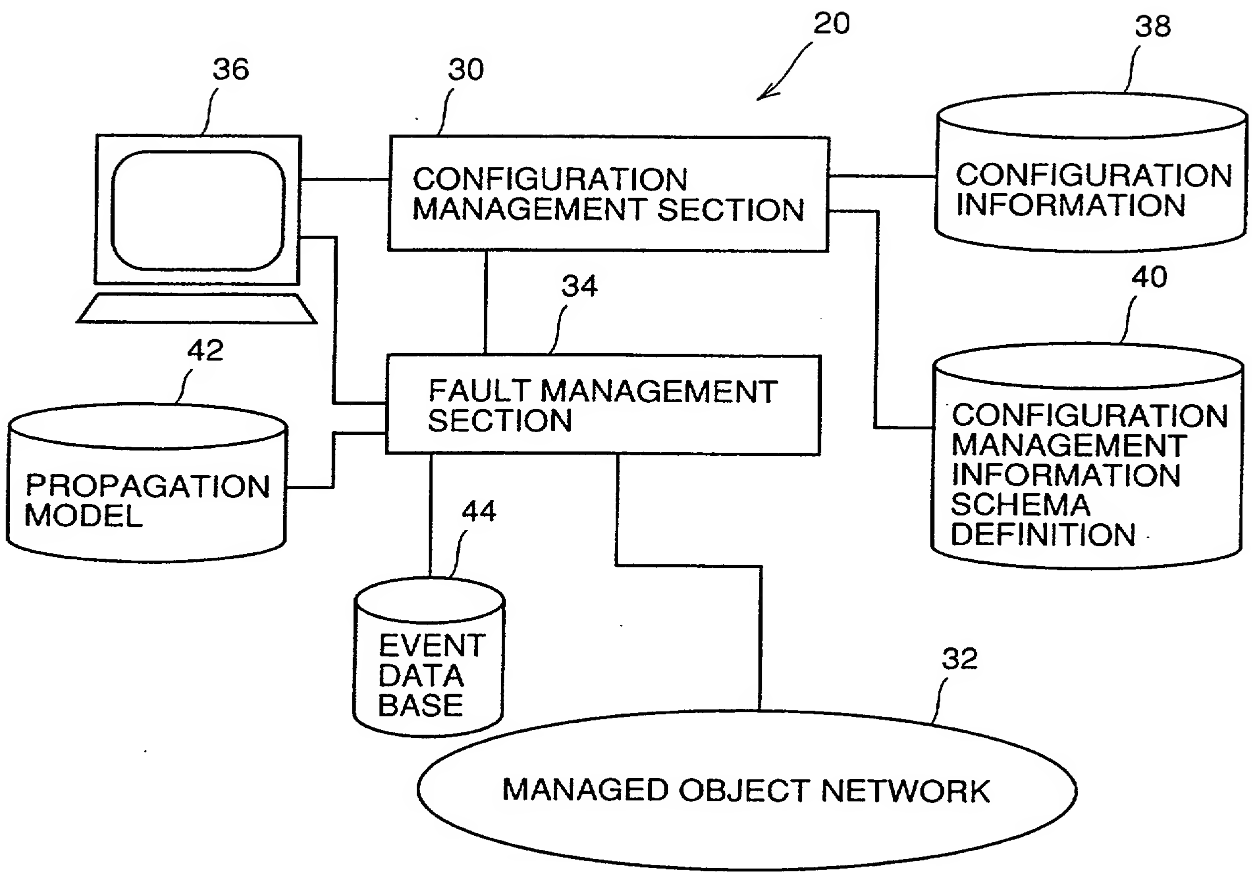
(57) A network managing system comprising a configuration management information schema definition storage unit for storing therein a configuration management information schema definition describing classes representing apparatuses in a network and failure events observed in the classes, a propagation model storage unit for storing therein propagation models between failure events, a configuration information storage unit for storing therein configuration information about the apparatuses actually installed in the network, and an inferring device for making inference based on the configuration management information schema definition, the propagation models, and the configuration information, including a counter for dealing with the failure events occurring in the apparatuses and a comparator for inferring the causes of the failure events based on the contents of the counter according to a predetermined method.



a ... TO USER INTERFACE UNIT 36
34 ... FAILURE MANAGEMENT SECTION
50 ... TIMER
52 ... COUNT COMPARATOR
54 ... FAILURE COUNTER
56 ... FAILURE COUNTER GENERATION SECTION
58 ... PROPAGATION MODEL SECTION
60 ... PROPAGATION RULE DETECTION SECTION
62 ... EVENT DATABASE SECTION

GB 2 363 286 A

FIG. 1



20
36
30
38
CONFIGURATION MANAGEMENT SECTION
CONFIGURATION INFORMATION
34
40
42
FAULT MANAGEMENT SECTION
CONFIGURATION MANAGEMENT INFORMATION SCHEMA DEFINITION
PROPAGATION MODEL
44
EVENT DATA BASE
32
MANAGED OBJECT NETWORK

FIG. 2

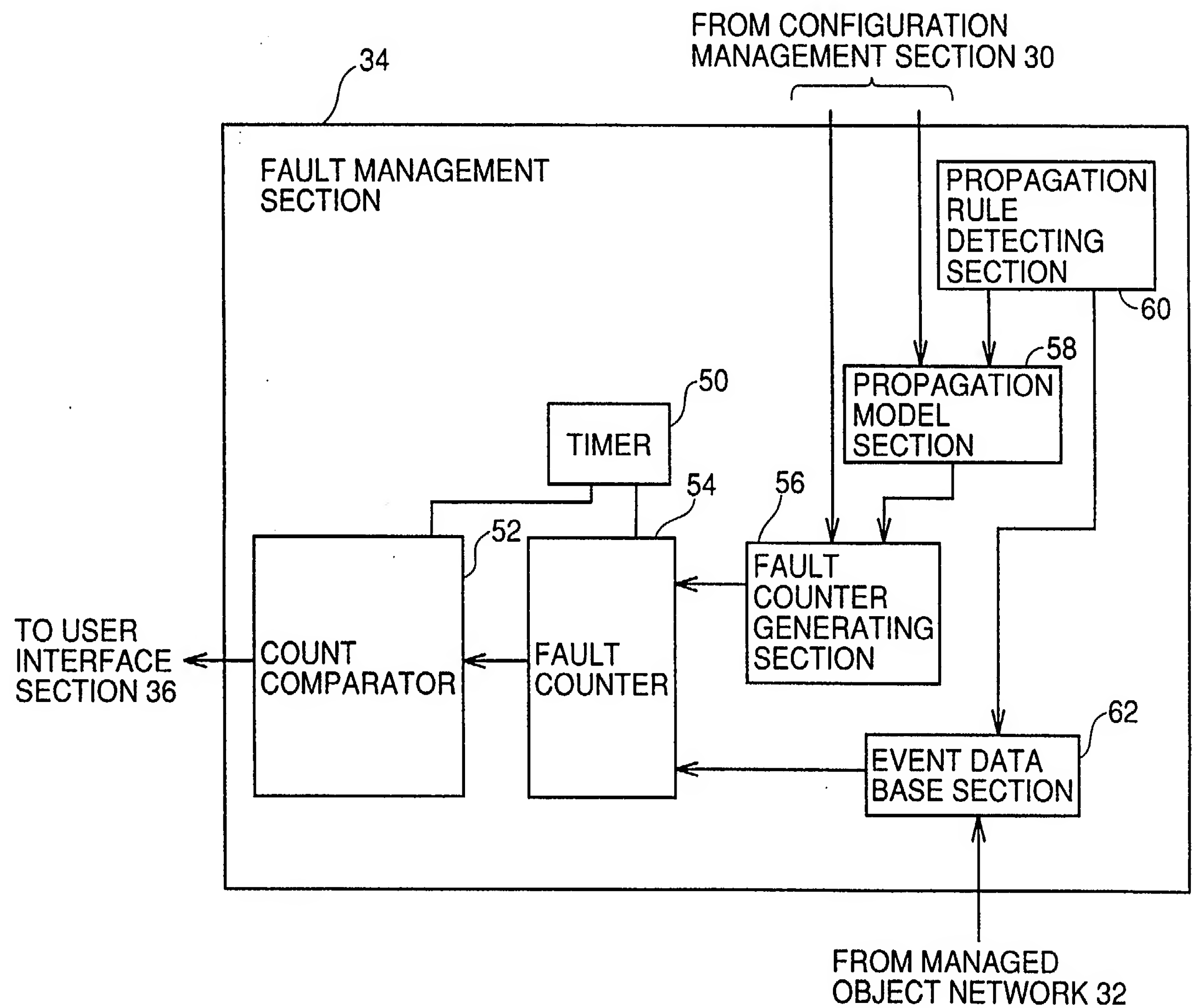

FROM CONFIGURATION
MANAGEMENT SECTION 30
34
FAULT MANAGEMENT
SECTION
PROPAGATION
RULE
DETECTING
SECTION
60
58
PROPAGATION
MODEL
SECTION
50
TIMER
54
56
52
FAULT
COUNTER
GENERATING
SECTION
TO USER
INTERFACE
SECTION 36
COUNT
COMPARATOR
FAULT
COUNTER
62
EVENT DATA
BASE SECTION
FROM MANAGED
OBJECT NETWORK 32

FIG. 3

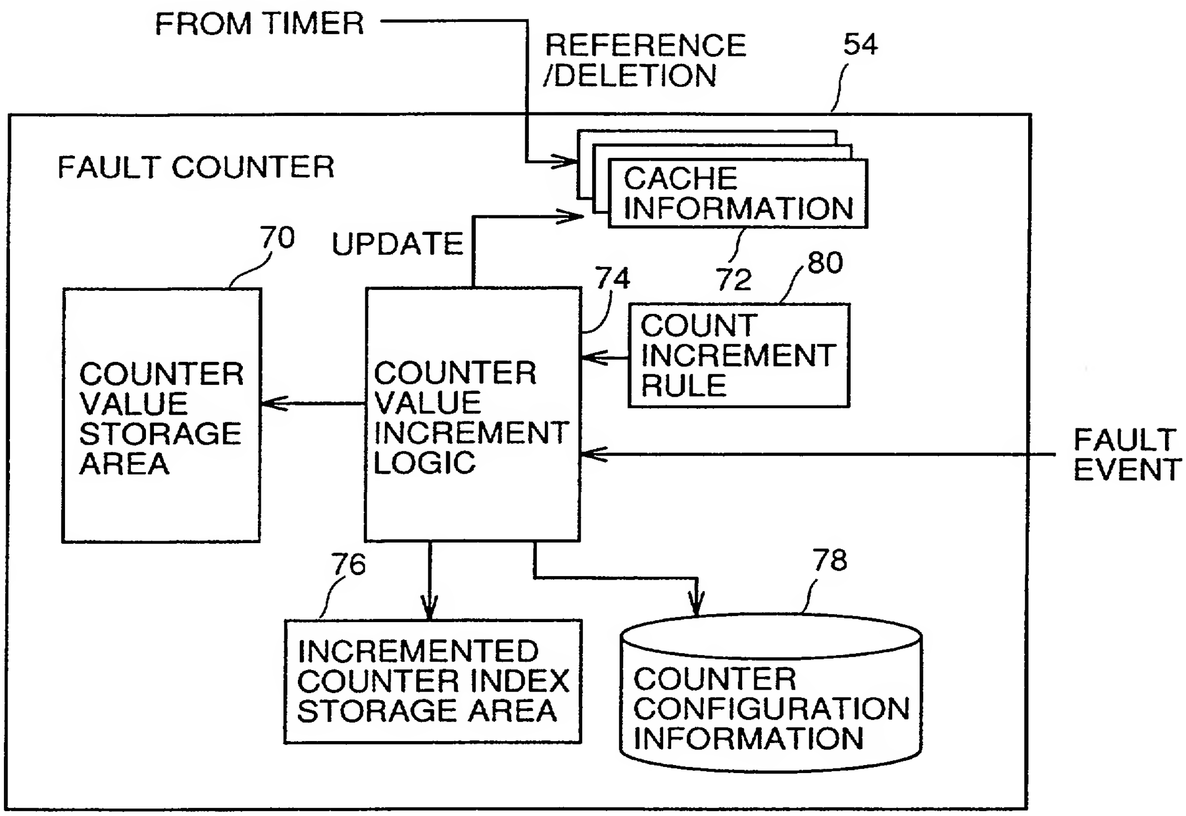

FROM TIMER
REFERENCE
/DELETION
54
FAULT COUNTER
CACHE
INFORMATION
70
UPDATE
74
72
80
COUNT
INCREMENT
RULE
COUNTER
VALUE
STORAGE
AREA
COUNTER
VALUE
INCREMENT
LOGIC
FAULT
EVENT
76
78
INCREMENTED
COUNTER INDEX
STORAGE AREA
COUNTER
CONFIGURATION
INFORMATION

FIG. 4

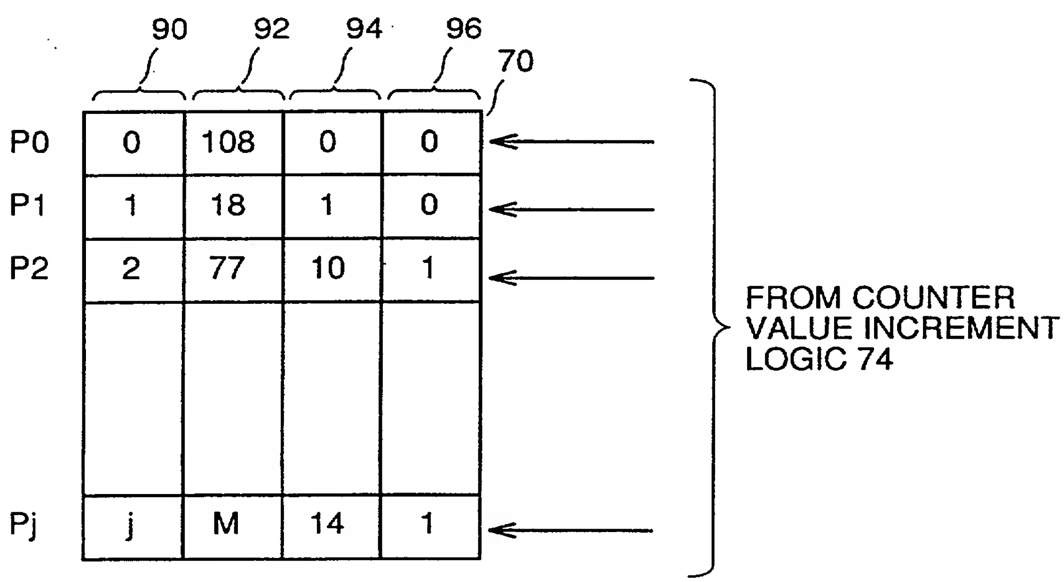

90
92
94
96
70
P0
0
108
0
0
P1
1
18
1
0
P2
2
77
10
1
Pj
j
M
14
1
FROM COUNTER
VALUE INCREMENT
LOGIC 74

FIG. 5

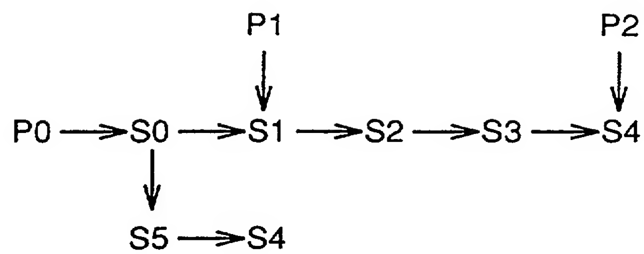


FIG. 6

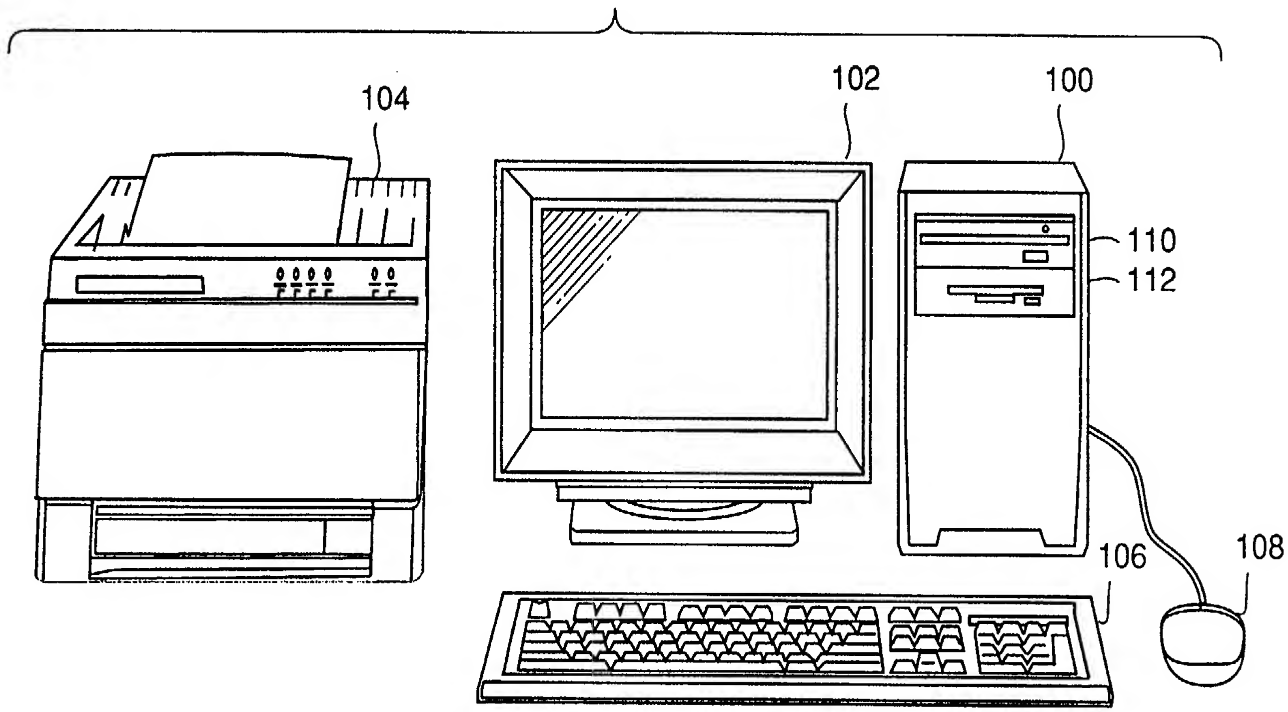


FIG. 7

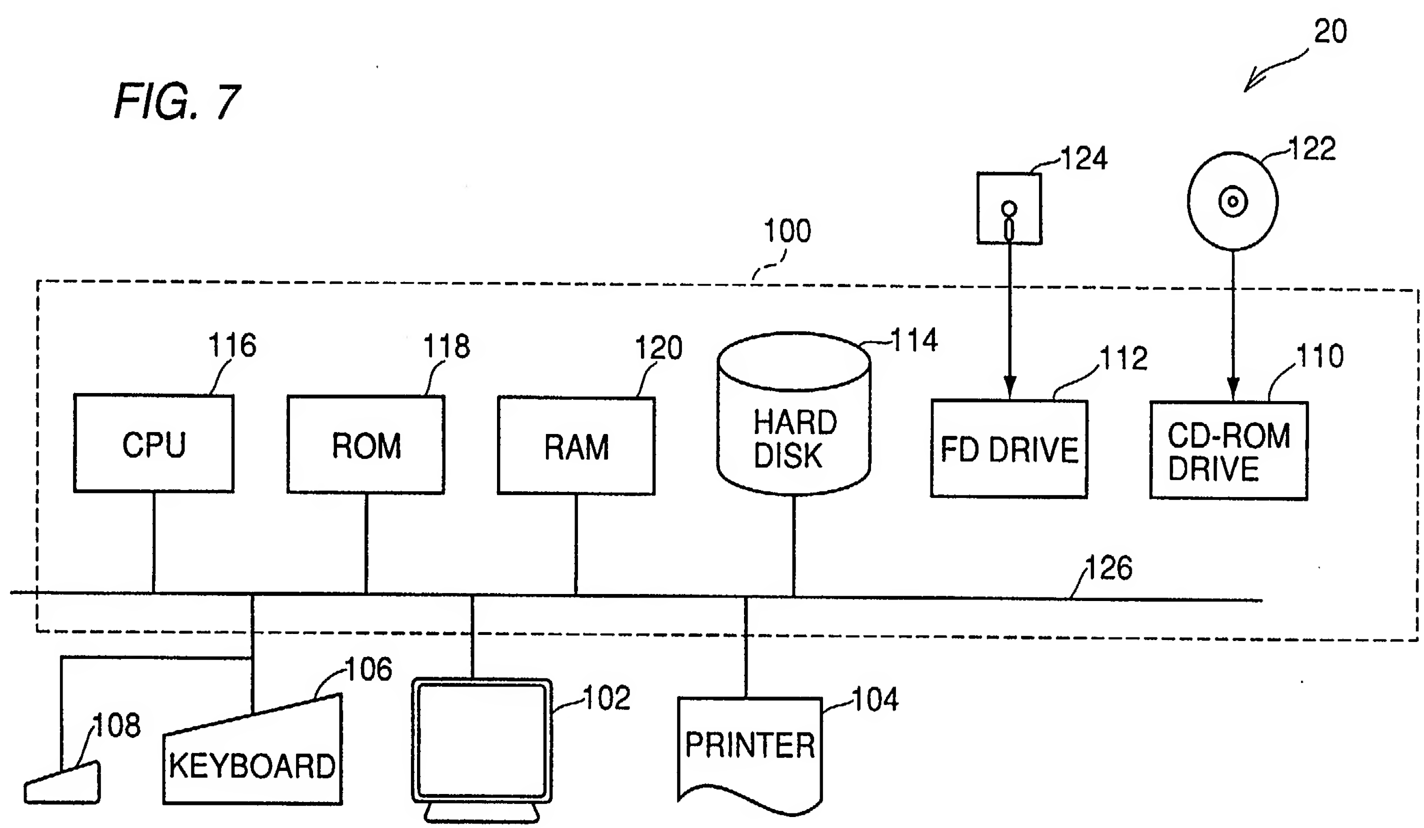

20
122
124
100
116
118
120
114
112
110
CPU
ROM
RAM
HARD DISK
FD DRIVE
CD-ROM DRIVE
126
106
108
102
104
KEYBOARD
PRINTER

FIG.8

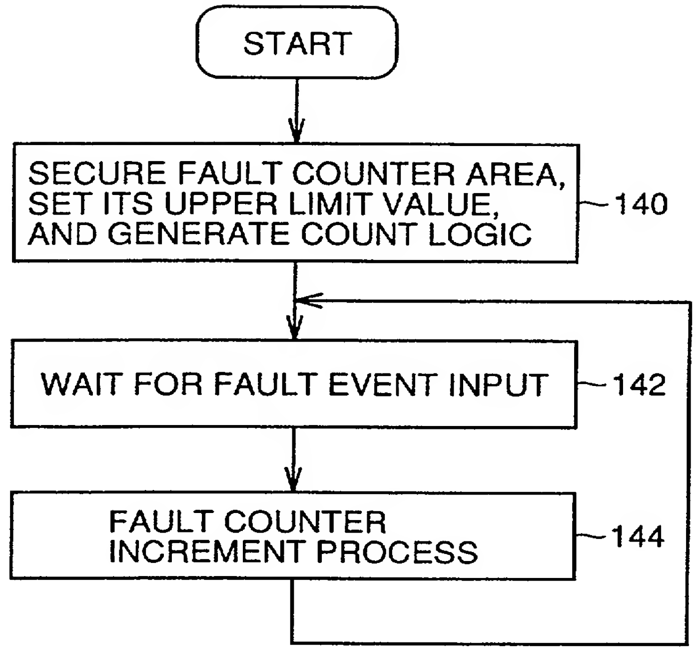

START
SECURE FAULT COUNTER AREA, SET ITS UPPER LIMIT VALUE, AND GENERATE COUNT LOGIC
140
WAIT FOR FAULT EVENT INPUT
142
FAULT COUNTER INCREMENT PROCESS
144

FIG.9

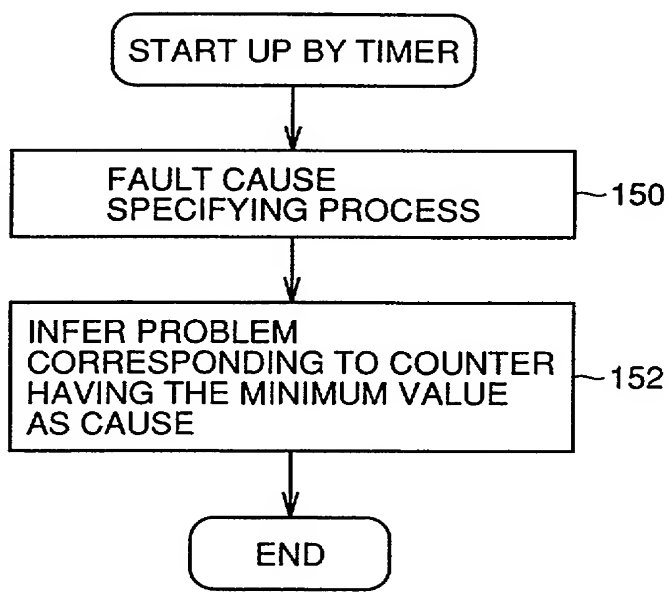

START UP BY TIMER
FAULT CAUSE SPECIFYING PROCESS
150
INFER PROBLEM CORRESPONDING TO COUNTER HAVING THE MINIMUM VALUE AS CAUSE
152
END

FIG. 10

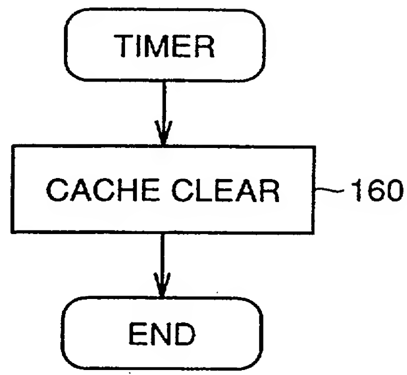
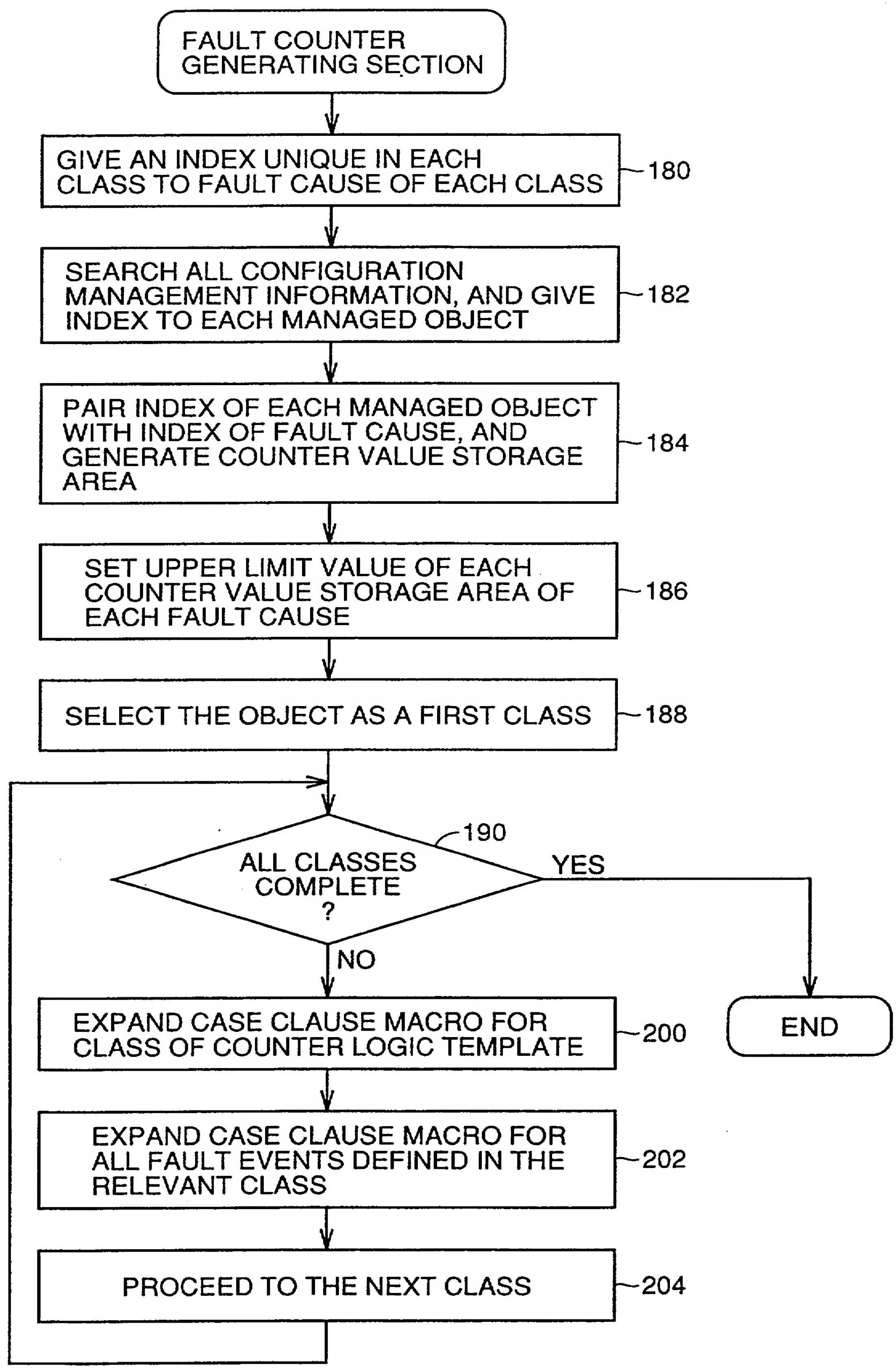


FIG.11



FAULT COUNTER GENERATING SECTION
GIVE AN INDEX UNIQUE IN EACH CLASS TO FAULT CAUSE OF EACH CLASS
180
SEARCH ALL CONFIGURATION MANAGEMENT INFORMATION, AND GIVE INDEX TO EACH MANAGED OBJECT
182
PAIR INDEX OF EACH MANAGED OBJECT WITH INDEX OF FAULT CAUSE, AND GENERATE COUNTER VALUE STORAGE AREA
184
SET UPPER LIMIT VALUE OF EACH COUNTER VALUE STORAGE AREA OF EACH FAULT CAUSE
186
SELECT THE OBJECT AS A FIRST CLASS
188
ALL CLASSES COMPLETE ?
190
YES
NO
EXPAND CASE CLAUSE MACRO FOR CLASS OF COUNTER LOGIC TEMPLATE
200
END
EXPAND CASE CLAUSE MACRO FOR ALL FAULT EVENTS DEFINED IN THE RELEVANT CLASS
202
PROCEED TO THE NEXT CLASS
204

FIG. 12

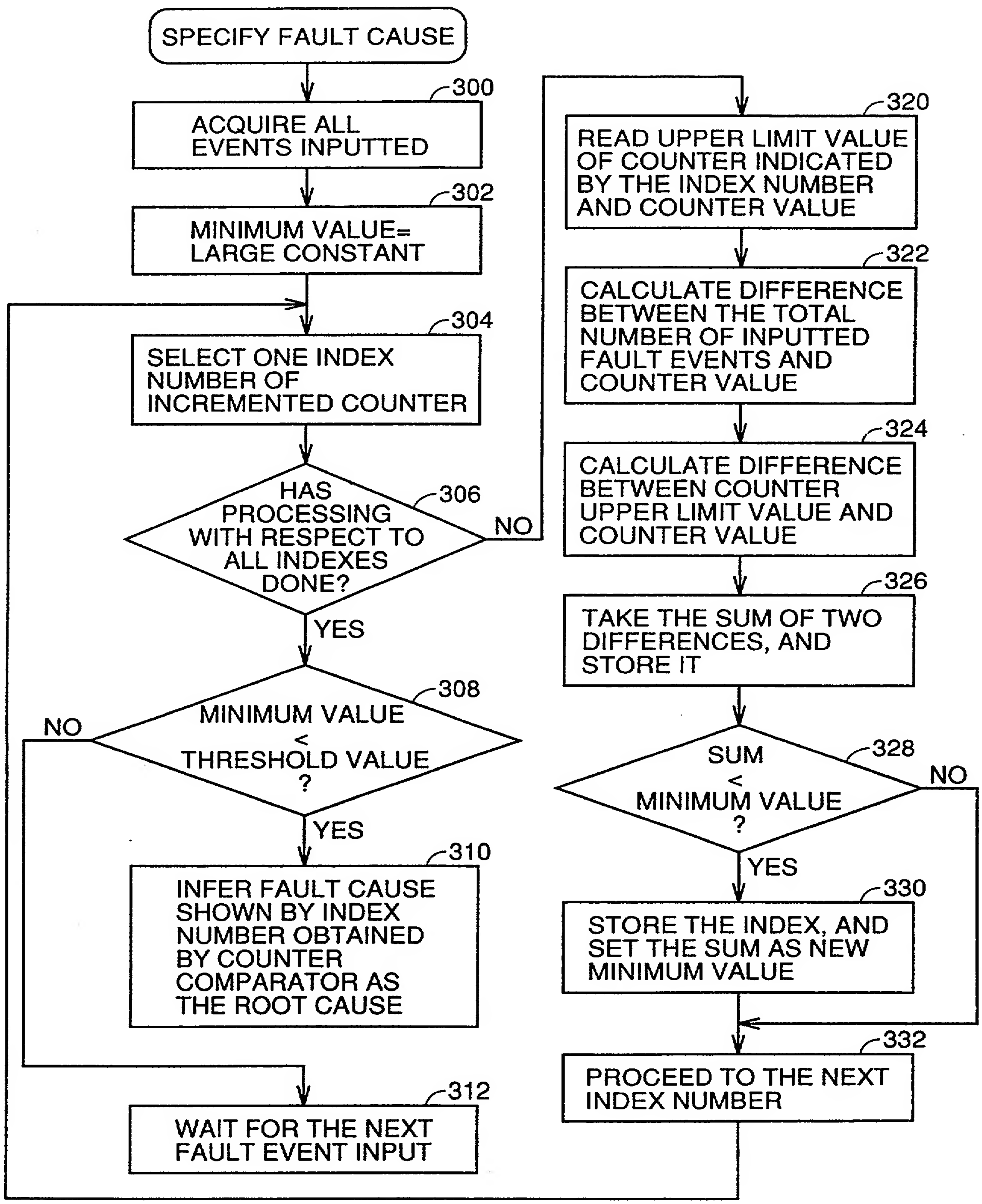

SPECIFY FAULT CAUSE
300
ACQUIRE ALL EVENTS INPUTTED
302
MINIMUM VALUE= LARGE CONSTANT
304
SELECT ONE INDEX NUMBER OF INCREMENTED COUNTER
306
HAS PROCESSING WITH RESPECT TO ALL INDEXES DONE?
NO
YES
308
MINIMUM VALUE < THRESHOLD VALUE ?
NO
YES
310
INFER FAULT CAUSE SHOWN BY INDEX NUMBER OBTAINED BY COUNTER COMPARATOR AS THE ROOT CAUSE
312
WAIT FOR THE NEXT FAULT EVENT INPUT
320
READ UPPER LIMIT VALUE OF COUNTER INDICATED BY THE INDEX NUMBER AND COUNTER VALUE
322
CALCULATE DIFFERENCE BETWEEN THE TOTAL NUMBER OF INPUTTED FAULT EVENTS AND COUNTER VALUE
324
CALCULATE DIFFERENCE BETWEEN COUNTER UPPER LIMIT VALUE AND COUNTER VALUE
326
TAKE THE SUM OF TWO DIFFERENCES, AND STORE IT
328
SUM < MINIMUM VALUE ?
NO
YES
330
STORE THE INDEX, AND SET THE SUM AS NEW MINIMUM VALUE
332
PROCEED TO THE NEXT INDEX NUMBER

FIG. 13

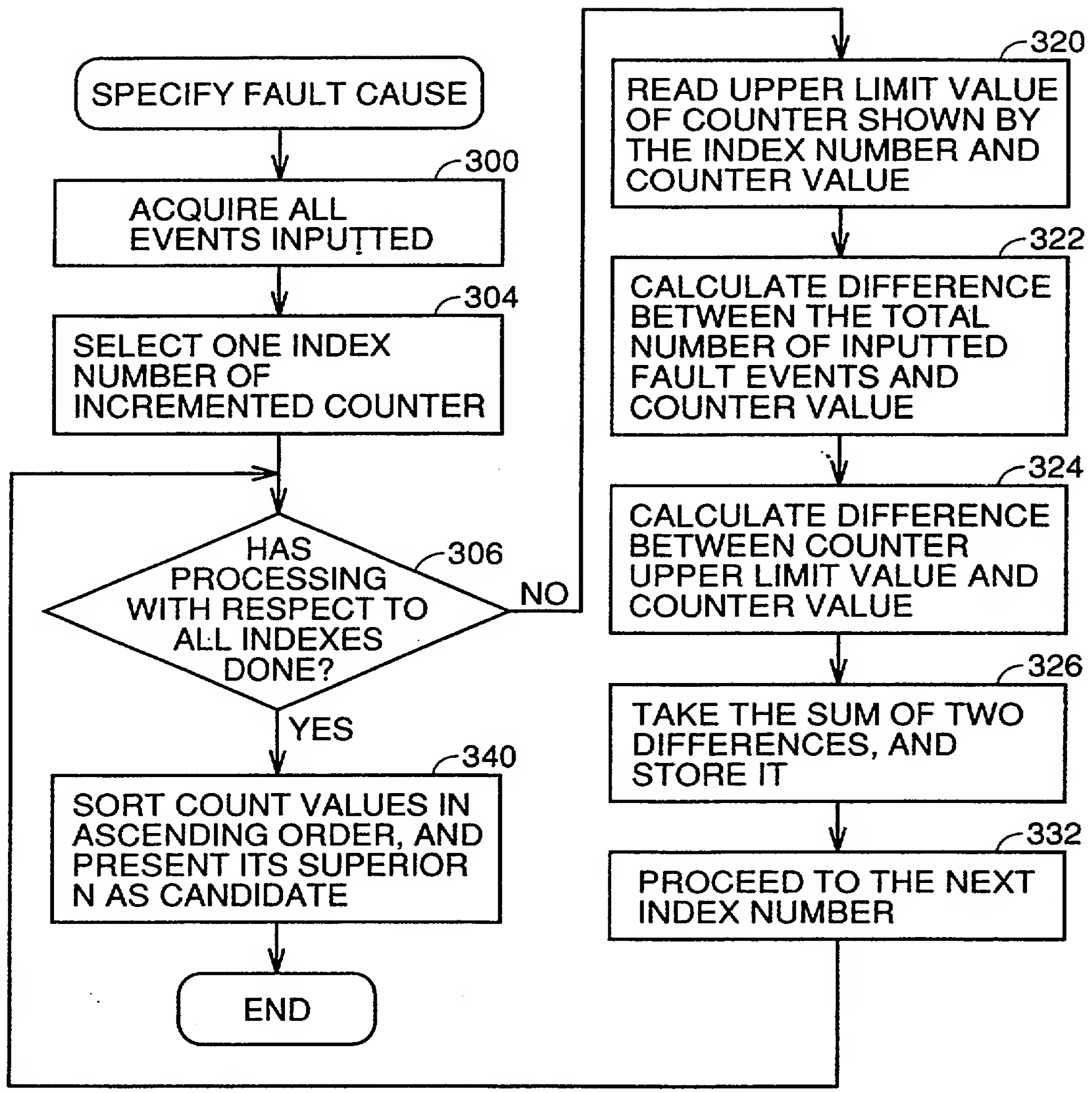

SPECIFY FAULT CAUSE
300
ACQUIRE ALL EVENTS INPUTTED
304
SELECT ONE INDEX NUMBER OF INCREMENTED COUNTER
306
HAS PROCESSING WITH RESPECT TO ALL INDEXES DONE?
NO
YES
340
SORT COUNT VALUES IN ASCENDING ORDER, AND PRESENT ITS SUPERIOR N AS CANDIDATE
END
320
READ UPPER LIMIT VALUE OF COUNTER SHOWN BY THE INDEX NUMBER AND COUNTER VALUE
322
CALCULATE DIFFERENCE BETWEEN THE TOTAL NUMBER OF INPUTTED FAULT EVENTS AND COUNTER VALUE
324
CALCULATE DIFFERENCE BETWEEN COUNTER UPPER LIMIT VALUE AND COUNTER VALUE
326
TAKE THE SUM OF TWO DIFFERENCES, AND STORE IT
332
PROCEED TO THE NEXT INDEX NUMBER

FIG. 14

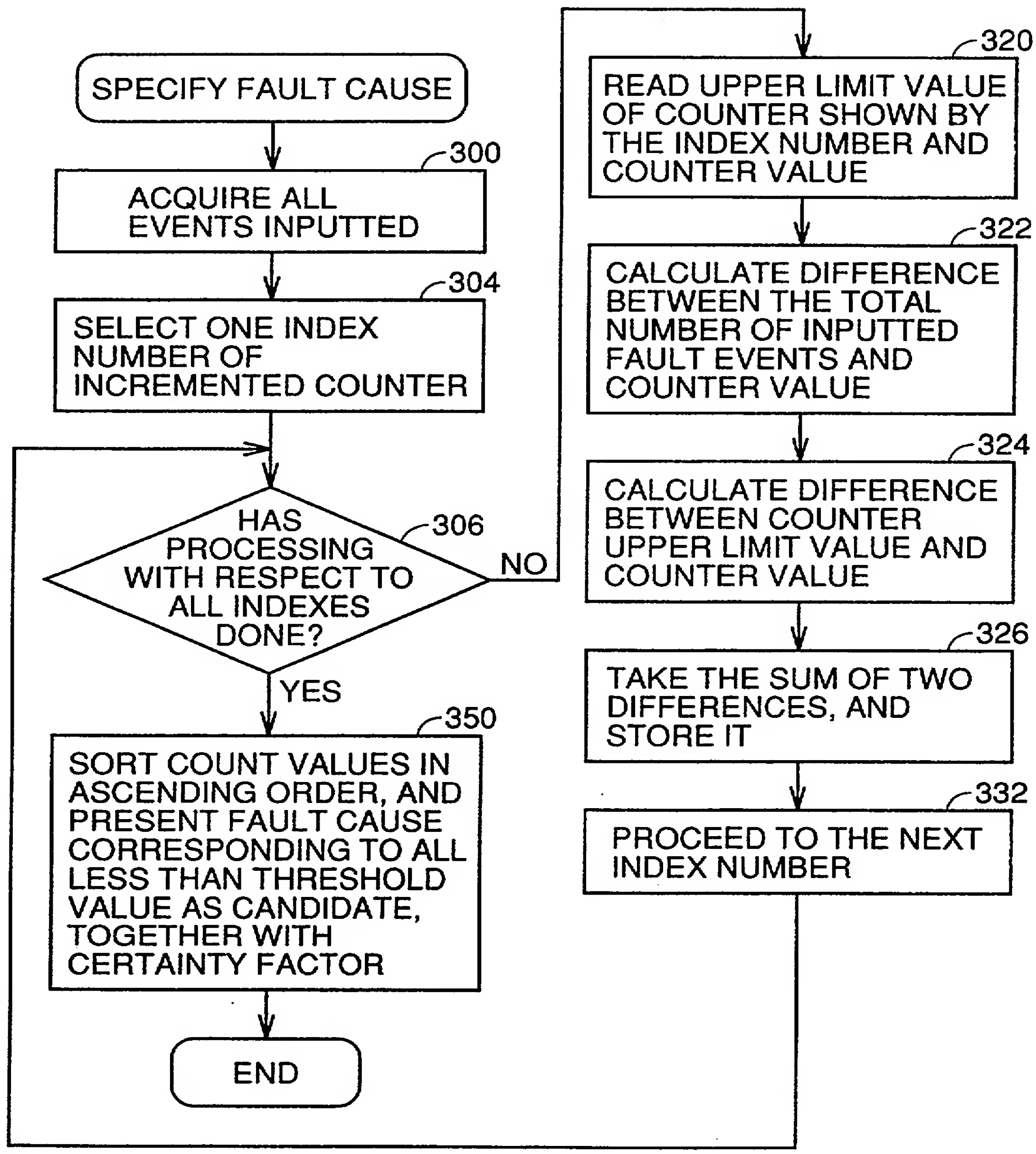
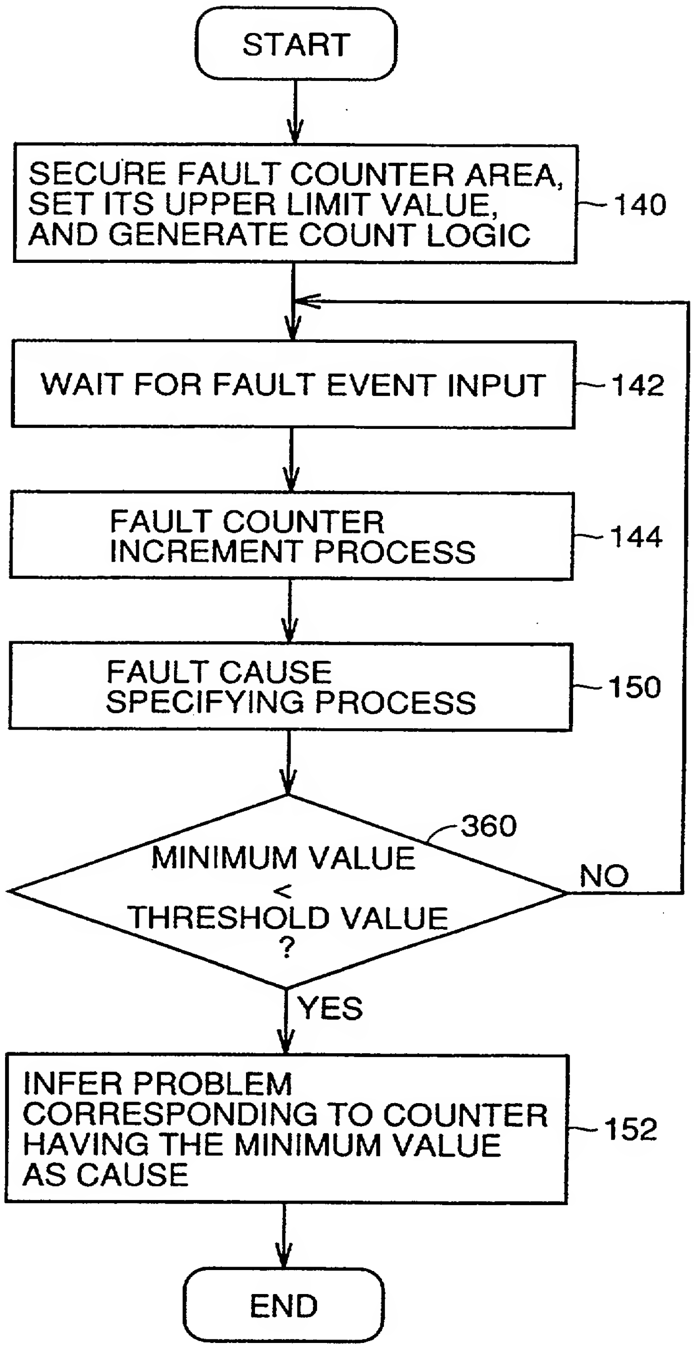

SPECIFY FAULT CAUSE
300
ACQUIRE ALL EVENTS INPUTTED
304
SELECT ONE INDEX NUMBER OF INCREMENTED COUNTER
306
HAS PROCESSING WITH RESPECT TO ALL INDEXES DONE?
NO
YES
320
READ UPPER LIMIT VALUE OF COUNTER SHOWN BY THE INDEX NUMBER AND COUNTER VALUE
322
CALCULATE DIFFERENCE BETWEEN THE TOTAL NUMBER OF INPUTTED FAULT EVENTS AND COUNTER VALUE
324
CALCULATE DIFFERENCE BETWEEN COUNTER UPPER LIMIT VALUE AND COUNTER VALUE
326
TAKE THE SUM OF TWO DIFFERENCES, AND STORE IT
332
PROCEED TO THE NEXT INDEX NUMBER
350
SORT COUNT VALUES IN ASCENDING ORDER, AND PRESENT FAULT CAUSE CORRESPONDING TO ALL LESS THAN THRESHOLD VALUE AS CANDIDATE, TOGETHER WITH CERTAINTY FACTOR
END

FIG. 15


START
SECURE FAULT COUNTER AREA, SET ITS UPPER LIMIT VALUE, AND GENERATE COUNT LOGIC
140
WAIT FOR FAULT EVENT INPUT
142
FAULT COUNTER INCREMENT PROCESS
144
FAULT CAUSE SPECIFYING PROCESS
150
MINIMUM VALUE < THRESHOLD VALUE ?
360
NO
YES
INFER PROBLEM CORRESPONDING TO COUNTER HAVING THE MINIMUM VALUE AS CAUSE
152
END

SPECIFICATION

NETWORK MANAGEMENT SYSTEM

Technical Field

The present invention generally relates to a network management system that manages a fault on a network, and more particularly, to a network management system that has a function of specifying a root cause of fault from various symptoms of a plurality of faults observed on a network.

Background Art

A large scale of communication network by computers is now developing. As the communication network comes to have larger scale, an influence by a fault generated on a network is also becoming great and serious. For this reason, it is very important to effectively manage a network. The following defines terms related to network management used in the present specification.

The term "Event" means an exceptional state generated in a network. More specifically, the "Event" includes a fault in hardware and software, stoppage, a bottleneck of performance, an inconsistency of network configuration, an unexpected result by insufficient design, a damage caused by malicious computer virus and the like.

The term "Disadvantage" has the same meaning as "Event" in the present specification.

The term "Symptom" means an observable event. Namely, the term "Symptom" is the same as the term "Symptom Event". For example, the term "Symptom" includes the events such as: "it always takes time to communicate with a certain destination A, and a re-transmission is required", "an illegal character (data error) is always generated with respect to a certain destination B", "reception acknowledgement never comes back from a certain destination C".

The term "Problem" means a root cause of fault. The problem is

not always observable. An example of the term "problem" is a breakage of transmitter of communication device, a disconnection of communication cable, or a shortage of capacity of communication line, etc. The term "fault cause" is also used with the same meaning as the term "Problem".

The term "Problem Event" has the same meaning as the "Problem".

The term "Object" means something having clear boundary and meaning with respect to a concept, an abstract, or a subject matter.

The term "Object Class" means a group of objects having the same kind of properties (attributes), a common behavior (operation), a common relation to other objects, and common meaning.

The term "Class" has the same meaning as the "Object Class".

The term "Object Instance" means a certain specific one object belonging to a certain object class. An "Object Instance" is also simply called as an "Instance".

One problem event in a network resource can cause many symptom events of a plurality of related resources. Among the problems, some are observable; however, in general, the problem event is not always observable. For this reason, there is a need of specifying a problem which is the root cause of fault, from a plurality of symptoms. In order to specify a problem of the root cause, a network administrator must be able to make a correlation between observed various symptom events and the problem.

However, when a network develops into a large scale, observed symptom events become innumerable. Moreover, a "causality", namely, which problem causes which symptom, becomes complicate. For this reason, it is almost impossible for a network administrator to manually specify a problem of the root cause of fault.

In order to accurately specify a problem of the root cause from symptom events of massive faults observed on the network as described above at a high speed, there have been proposed the conventional techniques (prior art) disclosed in USP No. 5,528,516 issued on June 18, 1996, "Apparatus and Method for Event Correlation and Problem

Reporting", and in USP No. 5,661,668 issued on August 26, 1997, "Apparatus and Method for Analyzing and Correlating Events in a System Using a Causality Matrix".

According to the aforesaid conventional techniques, managed object modeling and event propagation modeling are statically abstracted, and then, an object-oriented concept is introduced so as to effectively perform modeling. According to these methods, first, various managed objects are modeled into classes, and then, relations between classes are defined. Further, a certain event is modeled as being propagated along the relation between classes. Concerning the object-oriented technique, please refer to various textbooks related to an object-oriented technique.

On the basis of a class system determined in the above manner, a managed network is modeled. More specifically, a managed object in the network is abstracted as one instance of a certain class. Next, a network is modeled such that an event is propagated through the instances (managed object) according to a relation defined between the class to which the instance belongs and a class to which an other instance belongs. On the basis of the network modeled in the above manner, a correlation between a problem and symptoms is predetermined.

To this end, propagation rules of symptom events are prepared. These propagation rules define the relation according to which, a problem event of the root cause of fault propagates to a symptom event of fault, and in turn, the symptom event propagates to another symptom event. A set of the propagation rules is called as a propagation model.

The above events include an event which is a problem event and symptom event at the same time, and an event which is neither a problem event nor a symptom event. In the aforesaid propagation model (rules), event propagation is modeled such that each event propagates between instances along a relation defined between managed object classes.

In particular, USP No. 5,528,516 mentioned above relates to a

fault management function, and more particularly, to an event correlation table approach for inputting many fault symptoms generated in a network fault, and for specifying the root cause at a high speed.

USP No. 5,661,668 mentioned above relates to a technique for generating the event correlation table used in the aforesaid USP No. 5,528,516. More specifically, the method disclosed in the USP No. 5,661,668 includes the following steps of:

(1) describing classes representing network equipment, and fault events observed in these classes;

(2) describing a propagation model between these fault events;

(3) describing a configuration information of the actual network;

(4) generating a matrix called Causality Mapping, showing a correspondence of the fault events and their causes based on above information in (1) to (3); and

(5) storing the above Causality Mapping in a recording medium of a computer, and generating an event correlation table used in the USP No. 5,528,516.

As described above, if the event correlation table is previously generated, it is possible to specify a problem of the root cause of fault by a relatively simple work of comparing a symptom pattern at an actual fault with a symptom pattern in the event correlation table. Therefore, the aforesaid conventional technique seems to specify a problem of the root cause of fault very easily.

However, the above conventional technique still has the following problems to be solved.

More specifically, the event correlation table used by the above conventional technique requires a memory capacity proportional to number of Problems x number of Symptoms. For this reason, the above conventional technique is disadvantageous in that a mass of storage capacity is required when a network becomes large; as a result, there is a limit to applying the above conventional technique to a large scale network.

In order to solve the above problem, the aforesaid USP has

described that a calculation for the root cause is made on a real time basis by compressing the obtained event correlation table. However, in the conventional technique, it is unclear to what extent the event correlation table can be compressed. Depending upon the actual network configuration, it is anticipated that the event correlation table cannot be effectively compressed in some cases. There is also a possibility that an information to be acquired is lost from the event correlation table during the compression. In such a case, an inference accuracy is inherently reduced. In order to prevent a reduction of the inference accuracy, a compression rate must be kept constant. Thus, time that would be spent for inference is proportional to an event correlation table size.

If the event correlation table is used, in order to specify a problem, a calculation proportional to number of Problems x number of Symptoms is always required. For this reason, it is possible to advantageously complete the calculation in a predetermined time regardless of how many of symptom events are generated in a predetermined time. Conversely, however, even if the number of symptom events generated in a predetermined time is small, this means that the predetermined time must be spent for inference. Thus, it is desirable to more quickly carry out an inference if the number of symptom events is small.

Moreover, in specifying the above problem, it is desirable not only to prevent an error in inferring the fault cause, but also to prevent a true fault cause from escaping the inference result. Furthermore, preferably, when a plurality of candidates for fault cause are given, the network administrator suitably determines importance of the fault causes. Further, although it is preferable if the network administrator can make an inference of fault cause at any time when he/she desires, it is further preferable if a fault cause is presented when a network fault is generated and its cause is specified.

It is, therefore, an object of the present invention to provide a network management system which can effectively specify a problem with a small amount of memory capacity.

Another object of the present invention is to provide a network management system which can effectively specify a problem with a small amount of memory capacity and a small amount of calculations.

Still another object of the present invention is to provide a network management system which can effectively specify a problem with a small amount of memory capacity and a small amount of calculations, and can prevent a defective specifying of a root problem.

Disclosure of Invention

The present invention relates to a network management system, and to a machine readable recording medium recording a program for causing a computer to operate as the network management system. The network management system includes: a configuration management information schema definition storing device for storing a configuration management information schema definition describing classes expressing equipments on a network and fault events observed in these classes; a propagation model storing device for storing a propagation model between fault events; a configuration information storing device for storing a configuration information of actual equipments on a network; and an inference device including: a counter provided on the basis of the configuration management information schema definition, the propagation model and the configuration information, corresponding to respective fault events to be generated in each equipment, and a comparison device for inferring a fault cause of the fault event by a predetermined method on the basis of contents of the counter. The counter includes: a storage device having a plurality of storage areas for storing respective count values; a device for setting an upper limit value with respect to each storage area on the basis of propagation model; and a logic circuit for updating a value of the storage area corresponding to an inputted fault event on the basis of the configuration management information schema definition, the propagation model, the configuration information, and a predetermined rule for update, in response to the inputted fault event.

The counter updates the content of the storage area corresponding to each fault cause, whereby the inference device infers the fault cause on the basis of the contents of the storage area. The necessary storage area is proportional to the number of instances included in the configuration information; therefore, the network management system obviously has the advantage as compared with the event correlation table that requires a storage capacity proportional to a square of the number of instances.

Preferably, the counter further includes means for storing a configuration information for counting consisting only of equipment connective relation, and the logic circuit refers to the predetermined rule and the configuration information for counting in response to a fault event input, and determines and causes counter to operated in response to the fault event input. Thus, in the counter operation, the logic circuit can trace the event propagation if there exists equipment connective information; therefore, it is possible to achieve high speed processing.

Preferably, the counter generates the plurality of storage areas when the network management system starts up, and sets an upper limit value of each storage area. Thus, once the storage area is generated in start up, it is possible to quickly execute subsequent count processing steps by the counter with respect to the fault event.

Preferably, the counter secures a necessary storage area in response to the fault event input, and sets an upper limit value corresponding to the necessary storage area. It secures a necessary storage area as the necessity arises. Moreover, it is possible to effectively use the storage device without using a storage area which is not frequently used.

Preferably, the inference device infers a true fault cause in response to each fault event input. Even when a network administrator makes no request of inference, the inference device infers the fault cause if the fault event is generated. Thus, it is possible to timely make an inference independent from a request of the network

administrator. Moreover, as compared with the case where an inference is made every predetermined time, it is possible to reduce a time lag from the generation of fault to inference.

Preferably, the inference device infers a true fault cause whenever a predetermined time elapses. As an inference is made every predetermined time, it is possible to timely make an inference without a request of the network administrator.

Preferably, the inference device calculates a distance between each fault cause and a true fault cause defined on the basis of a count value corresponding to each fault cause, and presents a predetermined number as fault cause candidates, starting from the fault cause having the small distance. As the inference device presents a predetermined number as candidates starting from the fault cause having the small distance, the possibility of true fault cause escaping candidates for fault cause.

Preferably, the inference device calculates a distance between each fault cause and a true fault cause defined on the basis of count value corresponding to each fault cause, and presents only fault cause having a distance smaller than a predetermined threshold value as a fault cause candidate. The inference device presents only such a fault cause that has a distance smaller than a predetermined threshold value; therefore, it is possible to present a fault cause having a high possibility to the network administrator.

Preferably, the inference circuit presents the fault cause candidates in the order sorted according to each distance. The fault cause candidates are sorted according to each distance, and thereby, it is possible to present the candidates to the network administrator according to the order of the possibility of being the fault cause. Therefore, the network administrator checks a portion which is considered as a fault cause according to the above order, and thereby, it is possible to securely remove the fault.

Preferably, the inference device calculates and presents a certainty factor calculated in accordance with a count value of each

counter with respect to each presented fault cause candidate. Thus, by using the certainty factor, it is possible to intuitively grasp the possibility of fault cause.

Preferably, the counter further includes a counted area storage device for storing an information specifying a storage area updated by the logic device, and the inference device sets only that storage area which is stored in the counted area storage device as a calculating object in inference. The inference device does not set the storage area, which has not being updated, as a calculation object for inference. Thus, only the storage area requiring calculation is set as the calculation object; therefore, the calculation can be performed at a high speed.

Preferably, the storage device includes an area that stores a flag showing whether or not each storage area is updated in response to a predetermined fault event, and the logic circuit includes means for maintaining the flag, and determining whether or not a value in each storage area should be updated by the contents of the flag. There is a case where the identical event is generated from a certain cause via two or more causality routes (paths). By conversely tracing the above two or more routes with respect to the fault event input, there is a possibility that the corresponding storage area is updated two times or more with respect to the above certain cause. The calculation result will be incorrect, and a flag is provided so that the storage area updated with respect to a certain fault event is not to be updated any more. By the above processing step, it is possible to update the storage area without an error under the aforesaid causality.

Preferably, an update by the logic circuit includes a processing step of incrementing a value in each storage area of a cascading range of an inputted fault event by a predetermined value, and the predetermined value is a value larger than 0 and not larger than 1. The increment value is set to the above value, and thereby it is possible to suitably handle a fault propagation model according to which an event propagates with a certain probability.

Preferably, the counter further includes a cache device for storing

an information specifying the storage area updated by the logic circuit in response to a certain fault event input, and the logic circuit updates the storage area specified on the basis of the information stored in the cache device when the certain fault event is again inputted.

By storing the information specifying the storage area updated in response to the certain fault event input, when the same fault event is again inputted, it is possible to directly make an access and update the target storage area without tracing the propagation route. Therefore, it is possible to achieve high speed processing.

Preferably, the cache device stores an information indicative of a time when the information specifying the storage area is referred last time, and the counter further includes a device for deleting information not referred to for a predetermined time or more from the cache device. By deleting (erasing) an information, which has not been referred to for a predetermined time or more, from the cache device, it is possible to effectively use the storage area.

Preferably, the counter further includes propagation rule detecting device for storing an inputted fault event, detecting a cross correlation between the fault events, and feeding a propagation model of a new fault propagation which is not described in the previous propagation model, back to the logic circuit, and the logic circuit includes means that receives a feedback from the propagation rule detecting device, for rebuilding the predetermined rules. The inference rules are dynamically modified by a history of the actually generated fault event; therefore, it is possible to more accurately infer the fault cause.

Brief Description of Drawings

Fig. 1 is a block diagram showing a network management system according to one embodiment of the present invention;

Fig. 2 is a block diagram showing a fault management section 34 shown in Fig. 1;

Fig. 3 is a block diagram showing a fault counter 54 shown in Fig.

2;

Fig. 4 is an illustration schematically showing a construction of a counter value storage area 70 shown in Fig. 3;

Fig. 5 is an illustration explaining a conceptual loop of causality of cause and symptom;

Fig. 6 is a view showing an appearance of computer for realizing a network management system according to the present invention;

Fig. 7 is a block diagram showing a computer for realizing a network management system according to the present invention;

Fig. 8 is a flowchart showing a fault event increment process in one embodiment of the present invention;

Fig. 9 is a flowchart showing a fault cause specifying process started by a timer in one embodiment of the present invention;

Fig. 10 is a flowchart showing a cache clear process started by a timer in one embodiment of the present invention;

Fig. 11 is a flowchart showing a procedure of a fault counter generating section 56;

Fig. 12 is a flowchart showing a fault cause specifying process;

Fig. 13 is a flowchart showing another fault cause specifying process;

Fig. 14 is a flowchart showing a still another fault cause specifying process; and

Fig. 15 is a flowchart showing a fault counter increment process and a fault cause specifying process in another embodiment of the present invention.

Best Mode for Carrying Out the Invention

The reason why the problem arises in the prior art is that an event correlation table representing all relations between Problem and Symptom is used. According to the present invention, in order to solve the problem of the prior art, no event correlation table is prepared, and then, every time a symptom event is generated, a counter (called as "fault counter") provided with respect to a fault relative to the symptom

event is incremented on the basis of a propagation model, a managed object model, and an actual network configuration information. The number of propagation model is finite, and the types of symptom events are also finite. Thus, assuming that all symptom events were generated, each fault counter would be counted up to a certain determined number. Therefore, in the present invention, the above certain determined numbers are predetermined as upper limits of count numbers of the fault counter, and then, are compared to the number of the actually generated symptom events and the number of counts of each fault counter, and thereby, it is possible to specify a fault cause corresponding to a symptom event generation pattern.

Referring now to Fig. 1, a network management system 20 according to the present invention includes: a configuration management information schema definition 40 that stores (holds) a data base schema for defining a network configuration information; a configuration information section 38 that stores a network configuration information described using the above schema; a propagation model 42; a configuration management section 30 that receives an information from the configuration management information schema definition 40 and the configuration information section 38, and then, manages the network configuration information on the basis of these models and network configuration information; an event data base 44 that stores a configuration information data and a fault information data as an event; a fault management section 34 that collects a fault event of network, and infers a fault cause from the fault event; and a user interface section 36 that receives an information indicative of an inference result of a problem of the root cause of fault from the fault management section 34, and then, presents it to a network administrator. The configuration management information schema definition 40, configuration information section 38 and propagation model 38 are stored in a storage device such as a memory. The storage device storing these information may be the same, or may be separate ones.

Referring now to Fig. 2, the fault management section 34 includes:

an event data base section 62 that stores a fault event from a network in the event data base 44; a propagation model section 58 that manages and refers to a fault propagation rule (propagation model 42); a fault counter generating section 56 that generates a fault counter described later from a configuration information, its schema definition information and the propagation model 42; a fault counter 54 including a counter that is generated by the fault counter generating section 56, and is correspondent to each fault cause, incrementing a counter corresponding to a fault cause every time a fault event is inputted from the event data base section 62; a count comparator 52 that specifies a fault cause by carrying out a predetermined procedure with respect to a counter value of the fault counter 54, and supplies the specified fault cause to the user interface section 36; a timer 50 that periodically starts up the count comparator 52; and a propagation rule detecting section 60 that calculates a cross correlation between fault events from a history of the inputted fault event, and then, detects a fault propagation rule having, for example, a description omission, and further, feeds it back to the fault counter generating section 56 via the propagation model section 58.

Referring now to Fig. 3, the fault counter 54 includes: a counter configuration information 78 that only includes an information necessary for identifying an equipment type or equipment from a configuration information, and a connection information between equipments; a counter value storage area 70 that includes a counter array corresponding to a fault of each equipment generated from the propagation model 42 and the configuration information section 38; a counter value increment logic 74 that increments a counter corresponding to each fault of the counter value storage area 70; a counter increment rule 80 that describes an equipment type, a connection relation and which fault cause counter of equipment class should be incremented with respect to a fault event inputted to a counter, which are generated from the propagation model 42 when the counter value increment logic 74 refers thereto; an incremented counter index storage area 76 that is an area for recording an index of an

incremented fault counter; and a cache information 72 that stores (holds) a record which fault counter is incremented as the result when applying a certain rule to a fault event of a specific equipment.

Referring now to Fig. 4, the counter value storage area 70 includes an index 90 (consecutive numbers in this embodiment), an upper limit value 92 where the number of fault events which will be caused by a fault cause corresponding to the relevant counter on the propagation model is preset, a count value 94 that is an area for storing a count value, and an update flag 96 that is used for preventing a symptom (fault event) input from being accumulatively incremented when the identical symptom is generated from a certain fault cause via two or more causality paths, for each fault cause (problem P0, P1, P2, ...). These areas are maintained by the counter value increment logic 74.

Fig. 5 shows Causality Mapping for the case where the identical symptom is generated from a certain fault cause via two or more causality paths. Now, it is assumed that the following fault causality exists therein.

(1) A problem P0 causes symptoms S0 to S4 under the fault causality as shown in Fig. 5.

(2) A problem P1 causes symptoms S1 to S4 under the fault causality as shown in Fig. 5.

(3) A problem P2 causes the symptom S4.

In this case, a fault propagating from the symptom S0 to the symptom S5 causes the symptom S4; for this reason, a loop shown in the above Causality Mapping is formed in combination with the symptom S4 caused by the problem P2. More specifically, when the symptom S4 is inputted, by referring to the configuration information, first, a counter of the problem P2 is immediately incremented. Further, a counter of the problem P1 is incremented via a path of the symptoms of S4 —> S3 —> S2 —> S1. It is not determined whether the problem P0 is incremented via a path of the symptoms of S4 —> S3 —> S2 —> S1 —> S0 or a path of the symptoms of S4 —> S5 —> S0. However, the counter of the problem P0 must be prevented from being doubly

incremented when the symptom S4 is inputted. For this reason, in this embodiment, the problem P0 is provided with a flag showing whether an increment has been done for the symptom S4 input, and if the flag is set, the counter is not incremented.

Actually, the above network management system shown in Fig. 1 is realized by software executable on a computer such as a personal computer, a workstation or the like. Fig. 6 shows an external appearance of a computer system for realizing the network management system. Referring now to Fig. 6, the computer system includes a computer 100 having a CD-ROM (Compact Disc Read Only Memory) drive 110 and a FD (Flexible Disk) drive 112, a display 102, a printer 104, a keyboard 106, and a mouse 108.

Fig. 7 is a block diagram showing a construction of the computer. As shown in Fig. 7, the computer 100 configuring the network management system 20 includes a CPU (Central Processing Unit) 116, a ROM (Read Only Memory) 118, a RAM (Random Access Memory) 120, and a hard disk 114, in addition to the above CD-ROM drive 110 and the FD drive 112. A CD-ROM 122 is loaded in the CD-ROM drive 110 and a FD 124 in the FD drive 112.

As already described, the network management system is realized by computer hardware and software executed by the CPU 116. In general, such software is distributed stored on a storage medium such as a CD-ROM 122, a FD 124 and the like, and then, is read from the storage medium by the CD-ROM drive 110 or the FD drive 112, and thereafter, is temporarily stored in the hard disk 114. Further, the software is read from the hard disk 114 to the RAM 120, and then, is executed by the CPU 116.

The computer hardware shown in Fig. 6 and Fig. 7 is commonly known. Therefore, the most essential features of the present invention is software stored on the storage medium such as the CD-ROM 122, the FD 124, the hard disk 114 and the like. In this case, an operation of the computer shown in Fig. 6 and Fig. 7 is well known, and will be obvious to the person skilled in the art.

The following is a description on a structure of software configuring the network management system 20. In this embodiment, the software includes a process (see Fig. 8) for receiving a fault event after the device starts up and executing an increment of a fault counter, a process (see Fig. 9) for specifying a fault cause when the software is periodically invoked by the timer 50, and a process (see Fig. 10) for clearing a cache information 72 when the software is invoked by the timer.

Referring now to Fig. 8, a structure of the software for executing an increment of fault counter will be described below. When invoked, the software (device) first secures the counter value storage area 70, sets the upper limit value of the counter value storage area 70, and generates a counter value increment logic 74 (step 140). Then, the device waits for a fault event input (step 142), and when the fault event is inputted, executes a process for incrementing the fault counter (step 144), and again waits for a fault event input after execution (step 142). The details of these processes will be described later.

Referring now to Fig. 9, a fault cause specifying process 150 is periodically invoked by the timer 50. As a result of this process, the user interface section 36 presents a problem corresponding to a fault counter having the minimum value to a network administrator as the root cause of fault. This process corresponds to a process executed by the count comparator 52 shown in Fig. 2.

Referring now to Fig. 10, the cache information 72 is periodically cleared by the timer 50. A date when the cache is referred to last is stored in the cache information 72 as an attribute, and the timer 50 periodically checks the date so as to discard cached information that has not been referred to for a long time. By doing so, it is possible to save a storage capacity of the cache information 72, and to accommodate change in network configuration.

In the example shown in Fig. 8 to Fig. 10, the fault cause specifying process and the process for clearing the cache information 72 are invoked by the timer independently from the process for

incrementing the fault counter. However, the present invention is not limited to the above description. For example, the aforesaid processes may be executed in accordance with a request from the network administrator, or may be started up in response to a certain condition being satisfied in the process for incrementing the fault counter. Methods for realizing the above processes will be different from design to design; however, the method is inevitably determined according to requirements, and will be obvious to those skilled in the art.

Referring now to Fig. 11, a process (step 140) for generating a fault counter will be described below. This process is equivalent to a process by the fault counter generating section 56 shown in Fig. 2. First, an index unique in each class is given to every fault cause in every class (step 180). For example, this index corresponds to Application Down, Tcp Disconnect, Routing Error and the like, which will be described later.

Next, the whole configuration information is searched, and then, an index is given to each managed object (step 182). In the example described later, this is shown by managed objects mo1 and mo2.

Further, the index of each managed object and the index of each fault cause are paired with each other, and then, a counter value storage area is generated (step 184). More specifically, a certain fault counter in the counter value storage area 70 shown in Fig. 4 is specified by the managed object index and the fault cause index, and then, is accessed. The fault counters corresponding to the fault cause problems P0 and P1 shown in Fig. 4 are actually managed for each managed object, and in the case where the managed object is different even in the same fault, an independent counter is allocated to each problem.

A fault cascading range is evaluated, and the upper limit value is set (step 186) in each of the counter value storage areas of each fault cause.

The counter value increment logic 74 shown in Fig. 3 is generated in the following manner. Generation of the counter value increment logic 74 is executed by software. Prior to this process, the following

logic template is prepared.

(a) counter_logic_main

```
definition of necessary variables, etc. ...
while( true ){
     switch(classOf(mo1){
     //add case_clause_macro_for_class
     }
}
```

(b) case_clause_macro_for_class

```
case_clause_macro_for_class (ClassName)={
  case ClassName:
    switch(symptom1){
    //add case_clause_macro_for_symptom_name
    break;
  }
```

(c) case_clause_macro_for_symptom_name

```
case_clause_macro_for_symptom_name(ClassName, SymptomName,
                                    Relation,rClassName,
                                         ProblemName) = {
  case SymptomName:
    if(existCache(mo1, symptom1)){
      executeCache(mo1, symptom1);
      if(mo  !=mo1){
          appendCache(mo, symptom, mo1, symptom1);
      }
      return;
    }
    mo2  = search(mo1,Relation,  rClassName);
    if( mo2  ==NOT_FOUND) return;
    if( existCounter( mo2, ProblemName)  )  {
```

```
        counter [mo2][ProblemName].val ++;
        incrementDone(mo2, ProblemName);
        addCache(mo,  symptom, mo2, ProblemName);
      }
      symptom  = TcpDisconnect;  mo= mo2;
      break;
    }
```

The counter value increment logic 74 is generated in the following manner from the aforesaid macro templates, the propagation model 42, the configuration management information schema definition 40 and the configuration information section 38.

First, the first class is selected as a processing object (step 188). Next, it is determined as to whether or not all classes are processed (step 190). If the processing of all classes is completed, this routine ends. If there is a class that has not yet been processed, the control proceeds to step 200.

In step 200, the above item (b) is expanded on a position shown as "add case_clause_macro_for_class" in the counter logic template of the above item (a). In this case, the "ClassName" of the above item (b) is replaced with a processing object class name (e.g., "Application"). Thereafter, the additional position in this process is moved to follow the expanded portion.

Further, the above item (c) is expanded in succession on a position shown as "add case_clause_macro_for_symptom_name" of the above item (b) expanded in step 200 with respect to all fault events defined in the relevant class (step 202). In this case, however, the "ClassName" of macro of the item (c) is replaced with a class name of processing object, "SymptomName" is replaced with each fault event name defined in the class, "Relation" is replaced with a relation name defined in the propagation model relative to the class, "rClassName" is replaced with a class name related to the class in the "Relation", and "ProblemName" is replaced with a fault cause given in step 180 with respect to the class,

respectively. In the manner as described above, in step 202, a macro expansion of the above item (c) is made with respect to all of fault events defined in the relevant class.

Subsequently, the next class is processed (step 204). In step 190, when it is determined that processing has been completed with respect to all classes, the counter value increment logic 74, in which the above items (b) and (c) are expanded, is obtained with respect to all classes defined in the class definition. An example of the counter value logic is shown below.

(d) Example of counter value logic:

```
mo1 = mo;
symptom1 = symptom
while(true){
  switch(classOf(mo1){
  case   Application:
         switch(symptom1){
         case   Application Down:
           if(  existCache(mo1,  symptom1)){
             executeCache(mo1,  symptom1);
             if(mo!  = mo1){
               appendCache(mo,  symptom,  mo1, symptom1);
             }
             return;
           }
           mo2 = search(mo1,  Underly,  TcpNode);
           if(mo2 == NOT_FOUND)  return;
           if(existCounter(mo2,  TcpDisconnect){
             counter  [mo2][TcpDisconnect].val ++;
             incrementDone(mo2,  TcpDisconnect);
             addCache(mo,   symptom,  mo2, TcpDisconnect);
           }
           symptom1 = TcpDisconnect;  mo1 = mo2;
```

```
                break;
              }
              break;
          case   TcpNode:
            switch(symptoml){
              case   TcpDisconnect;
                if(  existCache(mol   symptoml))  {
                  executeCache(mol,  symptoml);
                  if(mo!  = mol){
                    appendCache(mo,  symptom,  mol,  symptoml);
                  }
                  return;
                }
                mo2 = search(mol,  ConnectedTo,   Router);
                if(mo2 == NOT_FOUND) return;
                if(  existCounter(mo2,    RoutingError))      {
                  counter  [mo2][RoutingError].val ++;
                  incrementDone(mo2, RoutingError);
                  addCache(mo,  symptom,  mo2,  RoutingError);
                }
                symptoml  = TcpDisconnect;  mol  = mo2;
                break;
          }
```

In this case, each of functions and variables has the following function and semantics.

(e) Functionality and semantics of functions and variables

(Class definition)

An instance of a class “Application” has a relation of “Underly” to an instance of a class “TcpNode”.

The instance of a class “TcpNode” has a relation of “ConnectedTo” to an instance of a class “Router”.

(Propagation model)

A fault “RoutingError” of class “Router” cascades to a fault “TcpDisconnect” of “TcpNode” “ConnectedTo” it.

The fault “TcpDisconnect” of “TcpNode” cascades to a fault “ApplicationDown” of a class “Application” that is “Underlying” thereto.

(Functions)

search();

This is a function of searching a managed object of a designated class having a designated relation with a designated managed object.

existCache();

This is a function to check whether or not there exists a cache related to a designated fault event of the designated managed object.

executeCache();

This is a function of incrementing a fault counter according to a cache data related to a designated fault event of a designated managed object.

addCache();

This is a function of recording an increment of a designated fault counter of the designated managed object as a cache data with respect to a designated fault event of the designated managed object.

appendCache();

This is a function of appending a cache data of a designated fault cause of the designated managed object to another cache data.

existCounter();

This is a function to check whether or not there exists a counter of the designated fault cause of the designated managed object.

incrementDone();

This is a function for recording an increment of the counter of the designated fault cause of the designated managed object.

classOf();

This is a function of finding a managed object class.

mo1, mo2:

These are work areas of a managed object index.

Symptom1;

This is a work area of fault event name.

mo;

This is a managed object name in which an inputted fault event is generated.

symptom;

This is a fault name of the inputted fault event.

counter;

This is a fault counter corresponding to each fault cause.

The contents described in the above items "(a) counter_logic_main" to "(e) Function and semantics of functions and variables" are one example in this embodiment. The expression may differ depending upon equipments used, an operating system, a programming language and the like. Moreover, depending upon the operating system, there may be a case where a program calls system routines. In this case, it is those routines that actually carry out processing and it is possible that the routines are not included in what is distributed as software for the network management system. In such a case, however, it is defined how the routines should be arranged. Even if software includes no such routines, the software is within the scope of the present invention so long as the clamed function is realized by the software.

Unlike the prior art disclosed in USP. No. 5,661,668, the present invention does not store and hold Causality Mapping, but stores and holds a counter value increment logic, a rule and a counter configuration information 78 including relations between equipments only.

Next, the following is a description on a fault counter increment process shown in steps 144 of Fig. 8. A fault event is inputted in time series to a counter. An equipment identifier and an equipment type information for specifying an equipment in an actual configuration information are added to the fault event as an attribute. When a fault event is inputted, the counter value increment logic 74 makes an inquiry with respect to the counter configuration information 78 so as to find a fault counter of a problem corresponding to a fault of the managed object

corresponding to the equipment, and then, increments its counter value. In the example shown in "(d) Example of counter value logic:", when "ApplicationDown" is inputted as a fault event, the counter logic finds a counter of "TcpDisconnect" of the managed object corresponding to the equipment "Tcp", and then, increments its counter value. Then, the counter logic stores an index of the incremented fault counter in the incremented counter index storage area 76.

Further, the above "TcpDisconnect" is an input of another propagation rule; for this reason, the counter logic again inputs the "TcpDisconnect", and then, searches a router connected to the "Tcp" from the counter configuration information 78. Thereafter, the counter logic increments a count of "RoutingError" of the router. In the manner as described above, the counter logic repeats a continuous increment of each fault counter. When there is nothing to be propagated, the process with respect to the fault event ends. In this case, the counter logic sets a flag corresponding to the fault event of the fault counter when incrementing a certain fault counter with respect to a certain fault event, and further, increments the fault counter only in the case where no flag is set.

The templates shown in "(b) case_clause_macro_for_class" and "(c) case_clause_macro_for_symptom_name" are used for realizing the above process by replacing words in the templates with a specific class name or the like, and by expanding them.

The network management system notifies the fault cause corresponding to a value having the least distance of the values thus calculated to a network administrator as a true fault cause candidate. The following is a description on a fault cause specifying process. Referring now to Fig. 12, when the fault cause specifying process is started, first, all events inputted are acquired (step 300). Further, as initial setting (initialization), a predetermined large constant (e.g., the maximum number capable of being stored in a work area) is substituted for a work area for the minimum value (step 302). As described below, this is a preparation for storing, every time the minimum value is found

in succession in the middle of calculation, a value as the minimum value of calculation thus far.

Thereafter, one index number is taken out from the indexes stored in the incremented counter index storage area 76 (step 304). Then, a determination is made as to whether or not processing is completed with respect to all index numbers stored in the incremented counter index storage area 76 (step 306). If processing is completed with respect to all index numbers, the control proceeds to step 308, of which details will be described later. In this case, the calculation described below is made with respect to only those indexes stored in the incremented counter index storage area 76, and not all indexes. Other indexes are not incremented; for this reason, there is no influence even if the calculation is omitted. Therefore, it is possible to reduce a computational complexity, and to achieve high speed processing.

If processing is not completed with respect to all index numbers, the upper limit value of a counter indicated by the index number selected in step 304 and its counter value are read (step 320). Then, a difference between the inputted number of all fault events and the read counter value is calculated (step 322). Further, a difference between the upper limit value of the counter and the counter value is calculated (step 324). The sum of two differences thus obtained is stored (step 326). In the network management system 20 of this embodiment, the sum thus calculated is set as a value indicative of a possibility of fault cause (distance). Of course, although various calculating methods other than the above-described one may be conceived, the method described in this embodiment is the simplest.

Thereafter, a determination is made whether or not the sum thus calculated is smaller than the minimum value stored so far (step 328). If the sum is larger than the minimum value, the next index number is selected as a processing subject (step 332), and then, the control returns back to step 304. If the sum is smaller than the minimum value, the index number corresponding to the sum is stored, and then, the sum is stored as a new minimum value (step 330). Subsequently, the next

index number is selected as a processing subject (step 332), and then, the control returns back to step 304.

In the manner as described above, when processing is completed with respect to all index numbers, according to the determination in step 306, the control proceeds to step 308. In this step 308, a determination is made as to whether or not the minimum value obtained from a series of the above processing is smaller than a predetermined threshold value. If the minimum value is smaller than the threshold value, a fault cause shown by the index number corresponding to the minimum value is inferred as the root cause of fault, and then, is presented to a network administrator via the user interface section 36.

If the minimum value is larger than the threshold value, this means that the root cause is not inferred on the basis of a sufficient ground. Therefore, the system waits for the next fault event input (step 312), or displays a message that it is impossible to specify the root cause, and then, this process ends.

When a certain event is inputted, the network management system 20 of this embodiment stores information indicating that which managed object is finally reached and which fault counter is incremented, in the cache information 72. For example, if the fault event S4 is inputted under the causality as shown in Fig. 5, the fault counters of the P0, P1 and P2 are finally incremented. The above information is stored in the cache information 72, and thus, when the fault event S4 is inputted next, it is possible to immediately increment the corresponding fault counter from the configuration information without tracing the causality, and to achieve high speed processing.

Moreover, the network management system 20 of this embodiment has the propagation rule detecting section 60; therefore, it is possible to detect a cross correlation between fault events stored in the event data base 44, and to feed (add) a new fault propagation back to the propagation model 42 if the new fault propagation, which is not described in the propagation model 42, is detected. On the basis of a new propagation model 42 updated in the above manner, the counter

value increment logic 74 of the fault counter and a counter increment rule 80 are rebuilt, and thereby, it is possible to more accurately infer a fault cause.

It should be noted that a cross correlation between fault events may be detected in the following manner.

For example, concerning two arbitrary events S1 and S2, based on a time period t1 during which the event S1 is existing, a time period t2 during which the event S2 is existing, and a time period t12 during which these two events are co-existing in a certain time window, a correlation value C (S1, S2) between these events is calculated from the following equation.

$$C(S1,S2)=\frac{\sum t12}{\sqrt{\sum t1}\sqrt{\sum t2}}$$

The above value C (S1, S2) is calculated between all fault events generated in the time window. When the above value exceeds a threshold value, it is inferred that there exists a propagation relation between these events. Of the propagation relation thus detected, a rule that has not been described in the propagation model 42 yet is given to the propagation model section 58, and thereafter, is fed back to the fault counter generating section 56. The fault counter generating section 56 rebuilds a fault counter and a rule for increment on the basis of a rule updated in the above manner, and thereby, it is possible to make a more accurate inference. Of course, the equation for calculating the cross correlation value is not limited to the above equation, and various equations may be employed.

When presenting the inference result of fault cause to a network administrator using the user interface section 36, for example, a certainty factor z can be calculated in the following manner. The certainty factor means that a certain fault cause is a true fault cause:

$$z=1-\frac{n}{m}$$

where, n is a number matching the actually inputted fault event of the fault events inferred from the fault cause, m is a larger value of

either the total number of inputted fault events or the maximum upper limit value of the counter. Alternatively, in place of the maximum upper limit value of the counter, the upper limit value set in another counter may be used.

The certainty factor z of the above equation assumes a value in the range from 0 to 100% (0 to 1). Further, the certainty factor z is added to a fault cause candidate, and then, is notified. By presenting the fault cause with the aforesaid certainty factor to a user, the user can intuitively and readily recognize a probability that the fault cause is true rather than in terms of degree of difference.

As described above, according to the present invention, by defining an information that can be described by a relatively little labor, that is, an information (class definition, propagation model) which does not depend upon a specific network configuration information, it is possible to mechanically obtain the specific network configuration information by a method of automatically finding it. Therefore, it is possible to alleviate the burden of generating an information necessary for fault management (fault counter 54). Moreover, it is possible to flexibly deal with a dynamic change in configuration information (addition and deletion of equipments).

The event correlation table approach requires a relatively large storage capacity. According to the present embodiment, it is possible to realize a fault cause inference method based on a compact logic obtained from a description relative to class and propagation model and configuration information, and further, a consumed memory capacity is proportional to the number of instances included in the configuration information. Therefore, this system is advantageous as compared with the event correlation table approach requiring a memory capacity of a size proportional to a square of the number of instances.

Moreover, in the present embodiment, the counter value is incremented at a timing when the fault event is notified. No calculation is made when no fault is generated. In addition, only fault counter relative to the generated fault event is incremented; therefore, it

is possible to reduce a calculation load in increment.

In the fault cause specifying process, a distance is calculated with respect to only that fault counter that has been incremented by a fault event generated in a predetermined period (time window). Thus, when the number of generated fault events is limited, it is possible to greatly reduce a computational complexity (calculation) for specifying the fault cause.

The network management system 20 of the aforesaid embodiment presents only one fault cause corresponding to the index number having the minimum distance calculated based on the counter value of the fault counter, to a network administrator. However, the present invention is not limited to this embodiment. For example, the distance (the sum of the difference between the total number of fault events and the counter value and the difference between the upper limit value of the counter and the counter value), may be calculated with respect to all index numbers, and then, the fault cause corresponding to N (N is an arbitrarily natural number) index number(s) of the superior entity (distance is small) may be presented to the network administrator in the ascending order of distances.

A flowchart of the above case is shown in Fig. 13. In Fig. 13, like reference numerals are used to designate the same processing step as Fig. 12, and the details are not repeated here. In Fig. 13, there is no need of obtaining the minimum value of distance; therefore, the processing steps relative to steps 302, 308, 328 and 330 of Fig. 12 are omitted. Moreover, in place of the processing step 310 of Fig. 12, a processing step 340 is added. In step 340, a distance is obtained with respect to all index numbers (the decision result of step 306 is YES), and thereafter, the counter value is sorted in an ascending order, and then, only N counter values of higher order are presented as a fault cause candidate.

In the manner as described above, a plurality of candidates are sorted and outputted in succession from a candidate of the smallest distance, and thereby, the network administrator can confirm the fault

cause candidate in succession from the most doubtful candidate. Therefore, the network administrator can effectively specify and delete the fault cause. Moreover, a possibility that a true fault cause is missed from the fault cause candidates can be reduced.

In this case, only fault cause candidates having a distance less than a specific threshold value may be displayed. A flowchart of the above case is shown in Fig. 14. In Fig. 14, like reference numerals are used to designate the same processing steps as Fig. 13, and the details are not repeated here. In the example shown in Fig. 14, in place of the processing step 340 shown in Fig. 13, a processing step 350 is added. In the processing step 350, the counter value is sorted in an ascending order, and thereafter, the fault causes corresponding to all the values below a predetermined threshold value are presented as candidates to the network administrator together with the aforesaid certainty factor. As described above, all candidates smaller than the predetermined threshold value are presented, and thereby, it is possible to present a fault cause having a high probability to be true, with respect to a plurality of fault causes, and thus, to prevent an omission from inference result. Further, the certainty factor is given, and thereby, it is possible to readily make a distinction between candidates significant and not significant among all candidates.

Unlike the above embodiment, every time the fault event is inputted, a process for inferring the fault cause may be employed. Fig. 15 shows a flowchart of the whole processing steps of the above case. In Fig. 15, like reference numerals are used to designate the same processing steps as those shown in Fig. 8 or Fig. 9, and the details are not repeated here.

In Fig. 15, after the fault counter increment process is completed with respect to the inputted fault event in Fig. 8, the fault cause specifying process (step 150) is immediately executed without waiting for the next fault event input. Then, if it is determined that the minimum value of the distance obtained as a result of the above processing is smaller than a predetermined threshold value, the fault

cause corresponding to the counter is inferred as the root cause of fault.

As described above, every time the fault event is inputted, the fault cause specifying process is executed, and thereby, the fault cause is advantageously presented to the network administrator directly when an information necessary for inferring a fault by the fault event input is complete. In comparison with the case of starting up the process for specifying the fault cause by using the timer, it is possible to obtain the effect that a time lag from the generation of fault to the fault inference becomes shorter.

In the example shown in Fig. 8, when the device is started up, all fault counter areas are secured, and its cascading range is predicted, and thus, the upper limit value of each counter is obtained. By doing so, when the fault event is generated, the fault counter corresponding to the fault event is quickly incremented. However, in the aforesaid system, a load on the device becomes high in start up, and also, start time becomes longer. Therefore, as an alternative, rather than executing the above process in start up, a necessary counter value storage area may be generated as the necessity arises, and then, the upper limit value of the counter may be set. In this case, before the fault event is inputted and the corresponding counter is incremented, a check is made as to whether or not there exists the counter. If there is no counter, the area is secured, and then, the process for setting the upper limit value may be executed.

By doing so, the counter value storage areas will not be generated with respect to those fault events that are hardly generated; therefore, a memory usage can be improved. Moreover, as described above, at a start up, a load on the device can be reduced and the device can start to work quickly.

Moreover, in the above embodiment, the incremental value has been set to 1.0. However, the present invention is not limited to this, and an arbitrary number in a range from 0 to 1.0 may be an incremental unit. By doing so, it is possible to express a probability of the generation of fault by the same logic as the above embodiment.

For example, if the propagation model states that a probability propagating from the fault event S3 to the fault event S4 is 0.5, the corresponding counter is incremented by 0.5 at the point of time. Further, in the case where the propagation model states that a probability propagating from the fault event S2 to the fault event S3 is 0.3, the incremental value is the product of these probabilities, that is, $0.5 \times 0.3 = 0.15$. In this manner, the network management system of the present invention can handle a probability of fault propagation.

As described above, the network management system according to the present invention has been described on the basis of the above embodiment. The present invention is not limited to the system described in the above embodiment. The scope of the present invention should be defined by the description of each claim. Components equipment to constituent components of the embodiment disclosed in the present specification should be included in the scope of right of the present invention.

Industrial Applicability

As is evident from the above description, in the network management system of the present invention, it is possible to specify a fault cause by a small memory capacity and computational complexity (calculation). Therefore, the network management system of the present invention is suitable for effectively specifying and solving a fault cause when a fault is generated in a large scale or complicate network.

CLAIMS

1. A network management system (20), comprising:

configuration management information schema definition storing means (40) for storing a configuration management information schema definition describing classes expressing equipments on a network and fault events observed in these classes;

propagation model storing means (42) for storing a propagation model between fault events;

configuration information storing means (38) for storing a configuration information of actual equipments on a network; and

inference means including count means (54) provided so as to correspond to each fault event generated in each equipment on the basis of the configuration management information schema definition, the propagation model and the configuration information, and comparison means (52) for inferring a fault cause of the fault event by a predetermined method on the basis of a content of the count means (54); wherein

the count means (54) includes

storage means (70) having a plurality of storage areas for storing each count value,

means (140) for setting an upper limit value with respect to each storage area on the basis of the configuration management information schema definition, the propagation model and the configuration information, and

logic means (74) for updating a value of the storage area corresponding to an inputted fault event on the basis of the configuration management information schema definition, the propagation model, the configuration information, and a predetermined rule (80) for update, in response to the inputted fault event.

2. The network management system (20) according to claim 1, wherein the count means (54) further includes means (78) for storing a

count configuration information consisting only of equipments connective relation, and

the logic means (74) includes means for referring to the predetermined rule (80) and the count configuration information (78) in response to a fault event input, and for determining and operating the count means to be operated in response to the fault event input.

3. The network management system (20) according to claim 1, wherein the count means (54) further includes means (140) for generating the plurality of storage areas when the network management system (20) starts up, and for setting an upper limit value of each storage area.

4. The network management system (20) according to claim 1, wherein the count means (54) further includes means (140) for securing the necessary storage area in response to the fault event input, and for setting an upper limit value corresponding to the necessary storage area.

5. The network management system (20) according to claim 1, wherein the inference means (34) includes means (Fig. 15; 150, 152) for inferring a true fault cause in response to each fault event input.

6. The network management system (20) according to claim 1, wherein the inference means (34) includes means (Fig. 9; 150, 152) for inferring a true fault cause every time a predetermined time elapses.

7. The network management system (20) according to claim 1, wherein the inference means (34) includes means (340) for calculating a distance between each fault cause and a true fault cause defined on the basis of a count value corresponding to each fault cause, and for presenting a predetermined number as fault cause candidates in succession from the fault cause having the smallest distance.

8. The network management system (20) according to claim 1, wherein the inference means (34) includes means (308, 310) for calculating a distance between each fault cause and a true fault cause defined on the basis of a count value corresponding to each fault cause, and for presenting only that fault cause which has a distance smaller than a predetermined threshold value as a fault cause candidate.

9. The network management system (20) according to claim 8, wherein the inference means (34) includes means (340, 350) for presenting fault cause candidates in the order sorted according to respective distances.

10. The network management system (20) according to claim 1, wherein the inference means (34) includes means (350) for calculating and presenting a certainty factor calculated in accordance with a count value of each count means (54) with respect to each presented fault cause candidate.

11. The network management system (20) according to claim 1, wherein the count means (54) includes counted area storage means (76) for storing an information specifying a storage area updated by the logic means (74), and

the inference means (34) sets only that storage area which is stored in the counted area storage means (76) as a calculating object in inference.

12. The network management system (20) according to claim 1, wherein the storage means (70) includes an area which stores a flag indicating whether or not each storage area is updated in response to a predetermined fault event, and

the logic means (74) includes means which maintains the flag, and determines whether or not a value in each storage area should be updated by contents of the flag.

13. The network management system (20) according to claim 1, wherein an update by the logic means (74) includes a processing step of incrementing a value in each storage area of a cascading range of an inputted fault event by a predetermined value, and

the predetermined value is larger than 0 and not larger than 1.

14. The network management system (20) according to claim 1, wherein the count means (54) further includes cache means (72) for storing an information specifying the storage area updated by the logic means (74) in response to a certain event input, and

the logic means (74) includes means for updating the storage area specified on the basis of the information stored in the cache means (72) when the certain fault event is again inputted.

15. The network management system (20) according to claim 14, wherein the cache means (72) stores an information indicative of a time when the information specifying the storage area is last referred, and

the count means (54) further includes means for deleting an information not referred to at least for a predetermined time from the cache means (72).

16. The network management system (20) according to claim 1, wherein the count means (54) further includes propagation rule detecting means (60) for storing an inputted fault event, detecting a cross correlation between the fault events, and feeding a propagation model of a new fault propagation which is not described in the previous propagation model, back to the logic means (74), and

the logic means (74) includes means which receives a feedback from the propagation rule detecting means (60), and rebuilds the predetermined rule.

17. A machine readable recording medium recording a program

for operating a computer as a network management system (20),

the network management system (20) comprising:

configuration management information schema definition storing means (40) for storing a configuration management information schema definition describing classes expressing equipments on a network and fault events observed in these classes;

propagation model storing means (42) for storing a propagation model between fault events;

configuration information storing means (38) for storing a configuration information of actual equipments on a network; and

inference means including: count means (54) provided so as to correspond to each fault event generated in each equipment on the basis of the configuration management information schema definition, the propagation model and the configuration information; and comparison means (52) for inferring a fault cause of the fault event by a predetermined method on the basis of a content of the count means (54),

the count means (54) including:

storage means (70) having a plurality of storage areas for storing each count value;

means (140) for setting an upper limit value with respect to each storage area on the basis of the configuration management information schema definition, the propagation model and the configuration information; and

logic means (74) for updating a value of the storage area corresponding to an inputted fault event on the basis of the configuration management information schema definition, the propagation model, the configuration information, and a predetermined rule (80) for update, in response to the inputted fault event.

18. The machine readable recording medium according to claim 17, wherein the count means (54) further includes means (78) for storing a count configuration information consisting only of equipments connective relation, and

the logic means (74) includes means (Fig. 15; 142, 144) for referring to the predetermined rule and the count configuration information in response to a fault event input, and for determining and operating the count means (54) to be operated in response to the fault event input.

19. The machine readable recording medium according to claim 17, wherein the count means (54) further includes means (140) for generating the plurality of storage areas when the network management system (20) starts up, and for setting an upper limit value of each storage area.

20. The machine readable recording medium according to claim 17, wherein the count means (54) further includes means for securing the necessary storage area in response to the fault event input, and for setting an upper limit value corresponding to the necessary storage area.

21. The machine readable recording medium according to claim 17, wherein the inference means (34) includes means for inferring a true fault cause in response to each fault event input.

22. The machine readable recording medium according to claim 17, wherein the inference means (34) includes means (Fig. 9) for inferring a true fault cause every time a predetermined time elapses.

23. The machine readable recording medium according to claim 17, wherein the inference means (34) includes means (340) for calculating a distance between each fault cause and a true fault cause defined on the basis of a count value corresponding to each fault cause, and for presenting a predetermined number as fault cause candidates in succession from the fault cause having the smallest distance.

24. The machine readable recording medium according to claim

17, wherein the inference means (34) includes means (308, 350) for calculating a distance between each fault cause and a true fault cause defined on the basis of a count value corresponding to each fault cause, and for presenting only that fault cause which has a distance smaller than a predetermined threshold value as a fault cause candidate.

25. The machine readable recording medium according to claim 24, wherein the inference means (34) includes means (340, 350) for presenting fault cause candidates in the order sorted according to respective distances.

26. The machine readable recording medium according to claim 17, wherein the inference means (34) includes means (350) for calculating and presenting a certainty factor calculated in accordance with a count value of each count means (54) with respect to each presented fault cause candidate.

27. The machine readable recording medium according to claim 17, wherein the count means (54) includes counted area storage means (76) for storing an information specifying a storage area updated by the logic means (74), and

the inference means (34) sets only that storage area which is stored in the counted area storage means (76) as a calculating object in inference.

28. The machine readable recording medium according to claim 17, wherein the storage means (70) includes an area which stores a flag indicating whether or not each storage area is updated in response to a predetermined fault event, and

the logic means (74) includes means which maintains the flag, and determines whether or not a value in each storage area should be updated by contents of the flag.

29. The machine readable recording medium according to claim 17, wherein an update by the logic means (74) includes a processing step of incrementing a value in each storage area of a cascading range of an inputted fault event by a predetermined value, and

the predetermined value is larger than 0 and not larger than 1.

30. The machine readable recording medium according to claim 17, wherein the count means (54) further includes cache means (72) for storing an information specifying the storage area updated by the logic means (74) in response to a certain event input, and

the logic means (74) includes means for updating the storage area specified on the basis of the information stored in the cache means (72) when the certain fault event is again inputted.

31. The machine readable recording medium according to claim 30, wherein the cache means (72) stores an information indicative of a time when the information specifying the storage area is last referred, and

the count means (54) further includes means for deleting an information having not referred to for at least a predetermined time from the cache means (72).

32. The machine readable recording medium according to claim 17, wherein the count means (54) further includes propagation rule detecting means (60) for storing an inputted fault event, detecting a cross correlation between the fault events, and feeding a propagation model of a new fault propagation which is not described in the previous propagation model, back to the logic means (74), and

the logic means (74) includes means which receives a feedback from the propagation rule detecting means (60), and rebuilds the predetermined rule.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP99/04041

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl^6 G06F13/00, 11/22, H04L12/24

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl^6 G06F13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Toroku Jitsuyo Shinan Koho	1994-1999
Kokai Jitsuyo Shinan Koho	1971-1999	Jitsuyo Shinan Toroku Koho	1996-1999

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP, 8-18593, A (IBM), 19 January, 1996 (19. 01. 96), Full text ; Figs. 1 to 5 & FR, 2722354, A & US, 5539877, A	1-32
Y	JP, 7-30540, A (Hitachi,Ltd.), 31 January, 1995 (31. 01. 95), Full text ; Figs. 1 to 12 (Family: none)	8-10, 16 24-26, 32
A	JP, 9-247145, A (NTT), 19 September, 1997 (19. 09. 97), Full text ; Figs. 1 to 9 (Family: none)	1-32

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 21 October, 1999 (21. 10. 99)	Date of mailing of the international search report 2 November, 1999 (02. 11. 99)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.